LA-UR- 04-3625

Approved for public release; distribution is unlimited.

Title:

RIA BEAM DYNAMICS DESIGN AND SIMULATION PROPOSAL: SCFY 021035

Author(s):

THOMAS WANGLER, LANSCE-1

Submitted to:

INTERNAL/EXTERNAL DISTRIBUTION

PROGRESS REPORT (FY2003)
RIA Beam Dynamics Design and Simulation
PROPOSAL: SCFY021035

May 6, 2004

Thomas P. Wangler[1], Robert W. Garnett[1], Dmitry Gorelov[3], Petr Ostroumov[4], Richard York[3], and Kenneth R. Crandall[2]

1 Los Alamos National Laboratory, Los Alamos, NM 87544
2 TechSource, Santa Fe, NM
3 Michigan State University, East Lansing, MI
4 Argonne National Laboratory, Argonne, IL

and **Advanced Beam Dynamics Modeling for RIA**

Robert Ryne[5], and Ji Qiang[5]
5 Lawrence Berkeley National Laboratory, Berkeley, CA

Introduction

Understanding beam losses is important for the high-intensity RIA driver linac. Small fractional beam losses can produce radioactivation of the beamline components that can cause radiation damage and can hinder hands-on maintenance, reducing facility availability. Operational and alignment errors in RIA can lead to beam losses caused by irreversible beam-emittance growth and halo formation. These issues impact the tolerance specifications and aperture choices in the linac and beam transport sections, as well as the design of the stripper regions; the simulations will extend from the low-energy beam transport (LEBT) line to the end of the linac. We have proposed a two year project to complete this work; the FY03 work described in this report represents the work done in the first year of the project. This progress report summarizes the FY03 funded work to develop a beam-dynamics simulation package for RIA driver linac simulations. The work was carried out between May, 2003 through February, 2004. The simulation package is based on modifications of two well-established accelerator design and simulation codes, PARMTEQ, and IMPACT. The PARMTEQ code has been modified so extensively that we have renamed this new RIA version as RIAPMTQ (pronounced RIA PARMTEQ). We intend that this simulation package will be particularly useful for computing the beam losses as well as for predicting the beam performance. The RIAPMTQ/IMPACT code package has been developed to run on parallel supercomputing platforms in anticipation of running simulations requiring large numbers of macroparticles for the beam-loss calculations. Low statistics runs can also be made on PC computers, which will be useful for the optimization work on the driver-linac design.

Overview

The present concept for the Rare Isotope Accelerator (RIA) project [1] includes a 1.4-GV superconducting driver linac operating with a 100% duty factor. The driver linac is designed for multicharge-state acceleration [2] of all stable species, including protons to 900 MeV and uranium to 400 MeV/u. In conventional heavy-ion linacs, a single charge-state beam of suitably high intensity is extracted from an electron-cyclotron resonance (ECR) ion source and injected into the linac. The linac typically contains one or more strippers at higher energies to further increase the beam charge state and improve acceleration efficiency. However, the limitation to a single charge state from the ion source and from each stripper significantly reduces the beam intensity. This disadvantage is addressed in the RIA driver-linac design concept by the innovative approach of simultaneous acceleration of multiple charge states of a given ion species, which results in high-power beams of several hundred kilowatts for all beams ranging from protons to uranium. Initial beam-dynamics studies [2], supported by experimental confirmation at the Argonne ATLAS facility [3], have demonstrated the feasibility of this new approach.

However, the high-power beam associated with the multiple charge-state acceleration introduces a new design constraint to control beam losses that cause radioactivation of the driver linac [4]. Radioactivation of the linac-beamline components will hinder routine maintenance and result in reduced availability of the facility. Therefore, it will be important for the RIA project to produce a robust beam-dynamics design of the driver linac that minimizes the threat of beam losses. As an important consequence of this design requirement, it will be necessary to develop a computer-simulation code with the capability of accurately modeling the beam dynamics throughout the linac and computing the beam losses, especially at high energies where beam loss translates into greater activation. To appreciate the scope of the requirements for this simulation code, we review the RIA driver-linac design concept.

The driver linac is made up of three sections. The first is the pre-stripper accelerator section consisting of 1) an ECR ion source, and 2) a low-energy beam transport (LEBT) line, which includes a mass and charge-state-selection system, and a buncher/RFQ injection system. This is followed by the initial linac stage consisting of 3) a room-temperature radiofrequency-quadrupole linac, a medium-energy beam transport (MEBT) line, and 5) the low-velocity (low-β) superconducting accelerating structures. The pre-stripper section accelerates the beam, consisting of two charge states for uranium, to an energy of about 10 MeV/u, where the beam passes through the first stripper and new charge states are produced.

The second section of the linac uses medium-β superconducting structures to accelerate the multicharge-state beam from the first to the second stripper at an energy of about 85 MeV/u. This medium-β section accelerates about five charge states for uranium. This is followed by the third and final section of the linac, which uses high-β superconducting structures to accelerate typically four charge-states for uranium to a final energy of 400 MeV/u.

Following each stripper is a magnetic chicane section for charge state selection and matching of the multicharge beams to the superconducting linac. The chicanes contain magnetic lenses for transverse focusing and RF structures for longitudinal focusing. The beam may be collimated in the chicane region, where some part of the halo that could lead to subsequent beam losses can be removed.

The overall performance of the driver linac is crucially dependent on the performance of the LEBT and RFQ. The LEBT is designed to focus, bunch, and inject two charge states for uranium into alternate longitudinal buckets of the RFQ. The LEBT RF buncher system consists of two main components. The first RF buncher cavity system (multiharmonic buncher) uses four harmonics and is designed to capture 80% of each charge state within the longitudinal acceptance of the RFQ. A second RF buncher cavity matches the velocity of each charge state to the design velocity of the RFQ. The two charge states for uranium are injected into the RFQ in such a way that adjacent RF buckets contain alternate charge states.

To avoid problems from beam-induced radioactivation, beam losses must be limited to low values, particularly in the high-energy part of the accelerator, where the beam loss rate must be less than about 1 watt per meter [5, 6]. The low beam-loss requirement imposes a challenge for controlling the emittance growth throughout the driver-linac, especially because of the complication of multiple charge-state beams. In addition to increasing the intensity, acceleration of multiple charge-state beams produces a larger total longitudinal emittance, increasing the threat of beam losses. For any proposed design it is imperative to compute the high-energy beam losses to ensure that the beam-loss requirements are satisfied. Such a computation normally requires the use of simulation codes that accurately track the beam particles through the whole accelerator using a physics model that includes all effects that can lead to emittance growth and possible beam losses. The development of such a simulation tool with relatively fast turnaround is the primary objective of our proposal.

Previous Simulation Code Developments for RIA

A significant amount of accelerator design work has already been done at two institutions, Argonne National Laboratory (ANL) [5] and Michigan State University (MSU) [7]. At present for superconducting linac simulations, the code LANA [7, 8] is used at MSU, and the code TRACK [9] is used at ANL. The LANA code was used extensively during the design and commissioning of the radioactive beam linac ISAC-1 at TRIUMF [10]. It was benchmarked as a result of the commissioning measurements, and is also being used for the design of ISAC-II, a superconducting linac for production of ion beams with energies above the Coulomb barrier.

These codes integrate the particle equations of motion through the electric and magnetic fields of the superconducting cavities, the 3D field components having been obtained from numerical solution of Maxwell's equations for each specified cavity geometry. The codes follow the motion of the particles in 6D phase space, and generally represent the dynamics of the multicharge-state beam with good spatial resolution. The codes are user friendly; they are written in Fortran language, have a modular structure, and have good graphical support for use on a PC. The codes also allow one to study effects on the beam caused by random errors associated with mechanical misalignments of focusing elements and rf cavities, and field errors, especially phase and amplitude errors in the rf cavities [5, 11, 12]. Recent work has begun at ANL [13] to develop an end-to-end multiprocessor code motivated by the same purpose as this project.

Although much code development has already taken place for RIA, more work to develop faster end-to-end simulation tools will be important for accurate computation of beam losses. In addition, the importance of demonstrating an understanding of the beam-losses justifies the development of more than one such code to provide the necessary cross checks on the simulations.

Development Plan for a New RIA Simulation Code

The problem of computing beam losses under realistic conditions requires the introduction of random machine errors including misalignments and field errors. The random error treatment is necessary for two reasons. First, in the as-built accelerator there are deviations of machine parameters from their exact design values, including errors in positioning of the beamline elements and field errors. Second, there are time fluctuations for some parameters, most notably the RF phase and amplitude of the superconducting cavities, which are driven by ambient mechanical vibrations (microphonics). This means that the problem of predicting the beam losses is inherently a statistical one. To be able to produce enough runs with a large enough number of particles, one needs to achieve fast turnaround. To accomplish this, we propose to develop a RIA beam-dynamics simulation-code package that runs in a parallel computing mode.

These simulation tools are important for computing beam losses for different RIA accelerator design options. At present an accelerator beam-dynamics design optimization process is already underway with several design options that are under development [5, 7, 14]. The simulation tools are important for the RIA project because they allow a fast and efficient capability for optimization of the machine design that properly addresses the beam-loss requirement. Since the beam-dynamics design must be completed at the earliest stages of the RIA project, this work addresses an important near-term need for the project.

We believe the quickest and most reliable way to produce the desired multiparticle simulation tools will be to use two existing simulation codes as the starting point. RIAPMTQ (pronounced RIA PARMTEQ) is based PARMTEQ [15], a Fortran code developed at LANL more than 20 years ago for simulation of low-energy ion beams in RFQs, and in low-energy beam transport lines both upstream and downstream of an RFQ. Thus, the PARMTEQ code can track particles through the low-energy beam transport (LEBT), the RFQ, and the medium-energy beam transport (MEBT). PARMTEQ provides an accurate RFQ model, calculating the beam dynamics in the RFQ using up to eight multipoles of the RFQ potential function. For beam simulations in the superconducting linac, we will use IMPACT (Integrated Map and Particle Accelerator Tracking Code)

[16], a Fortran code developed initially at LANL and more recently at LBNL. The IMPACT code already runs in parallel mode; the PARMTEQ code does not. IMPACT has been used to model the SNS linac, the CERN SPL, the KEK JPARC linac, the LEDA beam halo experiment at LANL, and in collaboration with physicists at GSI and CERN, for fundamental studies of intense beams.

The LEBT region is modeled with RIAPMTQ, and has a unique and rather complex set of requirements. The beam particles of both charge states are transported as a dc or continuous beam from the ion source to the buncher system. The design for uranium calls for two charge states, and both charge states must be matched and injected into the RFQ, so that alternate charge states are injected into adjacent buckets (separated by two cells). If desired, space-charge effects can be calculated by both RIAPMTQ and the IMPACT codes. We have assumed that space charge must be included in the LEBT and RFQ simulations at least, and the space-charge routine must be capable of handling both continuous beams and long bunches corresponding to beams in the process of being bunched. The standard version of PARMTEQ uses a 2D particle-in-cell space-charge subroutine called SCHEFF. SCHEFF has two versions, an X-Y version, which we are using, in the region of the LEBT where the beam is unbunched, and where the two charge states are horizontally separated, and the R-Z version, which is used where the beam is bunched or in the process of being bunched. These space charge subroutines are modified appropriately in RIAPMTQ to represent the transverse and longitudinal space-charge effects for beams with multiple charge states. Our initial plan is to modify RIAPMTQ to handle two charge states; initial simulations can begin at the charge-state selection slit in the LEBT, where the two charge states are selected.

For the beam-loss computation, statistical distributions of the beam are obtained by accumulating the results of repeated simulations with different initial random number seeds. Enough runs must be made for adequate statistical precision. This implies that one wants a relatively fast end-to-end simulation tool for rapid accumulation of the results with good statistical accuracy. Initially, we plan to do simulations with a few million particles per run. We plan to convert PARMTEQ to run on high performance computers, using parallel computing. Initially, we will run on the NERSC (National Energy Research Scientific Computing Center) computing facility at LBNL, where we already have familiarity and experience with that system.

The procedure we are adopting for the initial parallel computation is a relatively simple one called the particle decomposition method. Assuming that we are using N processors, the beam particles are initially distributed uniformly over these processors; each processor tracks the dynamics of its own subset of the particles. If the space-charge forces are not included in the simulation, the particles move independently, and the simulation can be performed as a fully parallel computation. The speed of the computation will be increased by nearly a factor N.

When the space-charge forces are turned on, all the particles interact; this affects the parallel computation effiency. To explain how the space-charge forces are handled in a parallel computation using the particle decomposition method, we can look at one cycle of the space-charge calculation. First, a spatial grid or mesh is superimposed on all the particles being tracked. Each processor has an identical copy of the full grid, and each deposits its particles on its copy. After this is completed, the accumulated charge distribution from each processor is made available to all the processors. All these distributions are added together by every processor in such a way that when this is completed, every processor has an identical copy of the total charge density on the full grid. Then, each processor solves the`Poisson equation using the full charge distribution to obtain the total electric potential and the total field components at each of the mesh points.

The next step is for each processor to interpolate the fields on the mesh points to the locations of the particles that belong to that processor. Then, each particle in each processor gets a kick and is advanced to a new position. The field interpolations and kicks are done in parallel by all the processors. A typical efficiency for parallel computation using this method is expected to be about 25 to 30%, when space charge is included, which is still a significant improvement in computational speed over that of a single processor. For example, with 128 processors, we anticipate that the computational speed would increase by about a factor of 32 to 42. The main advantage of this approach to parallel computation in the presence of space-charge forces is its simplicity. Ideally, the existing space-charge subroutine used for conventional serial computations, needs to be modified only by the introduction of a command that adds the charge distributions from each of the processors to accumulate the total charge distribution.

Status of RIAPMTQ Simulation Code Development for LEBT and RFQ

The work to develop RIAPMTQ has been divided into three steps. First, we have added the required multicharge-state physics changes to the standard version of PARMTEQ to create a RIAPMTQ that runs on our desktop computers. Second, we have installed the standard version of the PARMTEQ code on the NERSC supercomputer, and have modified it to run in parallel mode. The third step has been to insert the same physics changes already incorporated in the desktop computer version of RIAPMTQ into the modified parallel version of PARMTEQ that runs at NERSC.

The desktop version of RIAPMTQ has been modified to transport and accelerate beams with two charge states, including space charge, through the LEBT and RFQ. Fig. 1 shows RIAPMTQ simulation results with phase-space plots and rms sizes for a two-charge–state simulation run through a RIA LEBT and RFQ design. Space-charge forces have been included. In this simulation, the two-charge-state beam has been transported in the LEBT, bunched with a multiharmonic buncher, and accelerated to the end of the RIA RFQ.

In addition, we have installed the standard version of the PARMTEQ code at NERSC, and simulation results at NERSC have been benchmarked against the version that runs on a desktop computer. This work has involved many tasks for establishing compatibility, such as deleting the original Windows-based graphics commands, and modifying the I/O statements appropriate for use of ASCII format for all files. The next step for the RIAPMTQ version at NERSC was to insert the multicharge-state physics changes, and insert the commands required for parallel computing. In the second year of the project, we will carry out a systematic program of benchmarking of the rms results from the new RIAPMTQ against the existing TRACK and LANA codes.

Modifications for RIAPMTQ

To produce RIAPMTQ, we began with PARMTEQ, the basic documentation for which is available from the Los Alamos Accelerator Code Group [17]. The resulting RIAPMTQ tracks beam particles with either one or two charge states, first through a low energy beam transport line (LEBT), then through one or more RFQ structures, and a finally through a medium energy beam transport line (MEBT). For the RIA driver linac, the RIAPMTQ code is intended to model the linac from the charge state selection slit after the ion source to the beginning of the superconducting linac. (The superconducting linac is modeled by the IMPACT code.) Both the LEBT and the MEBT dynamics are modeled by subroutine PARTRAN, which can handle drift spaces, magnetic and electric quadrupoles, bending magnets, solenoids, and slits. The RFQ dynamics is modeled by subroutine RFQDYN. RIAPMTQ code uses longitudinal position or z as the independent variable. The length of the beam being tracked in the LEBT corresponds to one RF period at the fundamental frequency of the buncher system, which is one half the frequency of the RFQ. This corresponds to two adjacent periods of the RFQ, for which adjacent periods will have beams with different charge states.

In this section of the report, we discuss the changes that have been implemented to handle the multiple charge state dynamics for the RIA accelerator. The RIA driver linac is required to accelerate beams of any ions and multiple charge states. The specific example we have been using for the code development is uranium ions having charge states 28 and 29. The RFQ was designed for the average charge state, 28.5, at a frequency of 57.5 MHz. RIAPMTQ, like PARMTEQ, does not design the RFQ, but reads a file containing all of the required parameters.

PARMTEQ and RIAPMTQ are both used for simulating the beams through the Low-Energy Beam Transport (LEBT), the RFQ, and a transport system following the RFQ. In our design example the LEBT has several buncher cavities operated at harmonics or subharmonics of the RFQ frequency. The lowest buncher frequency is 28.75 MHz, half of the RFQ frequency, and this defines the "beam frequency". The design produces bunched beams entering the RFQ, the two charge states occupying adjacent acceleration buckets.

In the PARMTEQ input file (RFQ.IN), the following parameters have been added to make up a new LINAC line:

LINAC NT, W0, BFREQ, MASS, QSTATE, [QSTATE1, QSTATE2, CURRENT1, CURRENT2]

NT is the number of "tanks" in the RFQ (usually 1). W0 is the input energy of the design or reference particle, in either MeV or MeV/u, as will be explained below. BFREQ is the beam frequency in MHz. MASS is the mass number of the reference particle, to be entered as will be explained below. QSTATE is the charge state associated with the reference particle. QSTATE1 and QSTATE2 are the charge states of the two beams, and CURRENT1 and CURRENT2 are their respective electrical currents, in mA. A reference particle is needed because of the way we look at longitudinal phase space. We refer to $\Delta\phi$ and ΔW as being the phase and energy displacements of a particle from those of the reference particle.

There are two methods entering the input energy W0, mass number MASS, and charge states QSTATE, QSTATE1 and QSTATE2, and these quantities must be added in an internally consistent manner. If MASS is the mass number of the ion in amu (for example, 238 for uranium) , then W0 must be entered as the initial kinetic energy of this ion in MeV, and QSTATE, QSTATE1 and QSTATE2 must be entered as the actual charge states (28.5, 28 and 29). Alternatively, one may work in terms of energy per nucleon without needing to specify the specific mass number A, in which case one must set MASS = 1, the input energy W0 in MeV per nucleon, and the "charge states" must be entered as the charge-state per nucleon. For example, if the source voltage for uranium is 0.100 MV. A reference charge state of QSTATE= 28.5 would give an initial kinetic energy of 2.85 MeV, and the energy per nucleon would be 0.011975 MeV. The charge states for the RIA front end are QSTATE1=28 and QSTATE2=29, but in terms of charge states per nucleon, one must divide by A=238 to give QSTATE1=0.117647, and QSTATE2=0.121849. If the two uranium charge states together have a total current of 2 pμA, where "p" stands for "particle", this would mean almost 2 pμA X 28.5= 60 eμA (electrical). This implies the electrical currents of each beam should be about 0.030 mA. To operate in an MeV/amu beam energy mode, the LINAC card reads.

```
LINAC 1 0.01198 28.75 1. 0.11975 0.11765 0.12185 0.030 0.030
```

To operate in an MeV mode instead, rather than MeV/amu, the LINAC card reads

```
LINAC 1 2.85 28.75 238 28.5 28. 29. 0.030 0.030
```

You are not required to run two charge states. You may run only the reference beam, or only one charge state. Before any data is read, QSTATE and QSTATE1 are set to 1.0, QSTATE2 is set to zero, and CURRENT1 and CURRENT2 are set to zero. After the LINAC line is read, if QSTATE1, QSTATE2, CURRENT1 and CURRENT2 are omitted, QSTATE1 is set equal to QSTATE.

The numbers of particles in the two possible charge states are stored in NP1 and NP2, which are initialized to zero. These values are set after particles are generated by subroutine INPUTT. The first INPUT line will generate particles for charge state 1, and NP1 is set to NPOINT, specified as the second numerical entry on the INPUT line as the number of particles to be generated. These particle coordinates will be stored in the CORD array in CORD (I, 1) through CORD (I, NP1), where I is the index that identifies one the 6 phase space coordinates, and NPOINT=NP1 is the number of particles in this group, specified on the INPUT line. A second INPUT line (if there is one) will indicate that particles for a second charge state are to be generated. These particles will be stored in the CORD array in CORD (I, NP1+1) through CORD (I, NP1+NPOINT). After returning from subroutine

INPUTT the second time, NP2 is set to NPOINT, and NPOINT is then set to NP1 + NP2, the total number of particles in both charge states.

The following lines show how two charge states are generated, each having 2000 particles.

```
input -6 2000  0.000  60 .00980  0.000  60.0 .00980  360.  0.
0  0  0  0  0  -0.00021
input -6 2000  0.000  60 .00980  0.000  60.0 .00980  360.  0.
0  0  0  0  0   0.00021
```

The input parameters are the same, except for the initial energies, which are shifted from the reference energy per nucleon by the sixth parameter on the second line of each input. (In the case the shift is 0.5/238.) After returning from the INPUTT routine the second time, NP1 = 2000, NP2 = 2000, and NPOINT = 4000, which becomes the total number of particles.

When the particles are in an electric or magnetic field, the impulse they feel depends on their charge state. In these situations, before every DO loop over the total number of particles (NPOINT), the charge state, QS, is initialized to QSTATE1. At the end of the DO loop, the particle index (NP) is checked to see if it is equal to NP1. If so, and if NP2 > 0, QS is set equal to QSTATE2. This is not necessary in field-free regions where the charge does not affect the motion, such as in drift spaces, or when the particles are being transformed to another frame of reference, as in misaligned elements.

Three additional COMMON blocks are used to share the parameters required for the dynamics. One common block is used for the reference parameters.

```
common /reference/bfreq,qref,Wref,phiref,betar,bgr
```

Bfreq is the beam frequency in MHz, determined by the lowest frequency of the bunchers; qref is the charge state (in this case, per nucleon) of the reference particle; Wref is the energy per nucleon in MeV/amu of the reference particle; phiref is the design phase of the reference particle in radians; and betar and bgr are the β and $\beta\gamma$ of the reference particle.

Another common block has the charge states per nucleon, electrical currents in mA and number of particles in each of the two beams.

```
common /charg/ qstate1,  current1, qstate2, current2, np1, np2
```

The third common block is used for storing the total distance (tdist in meters); an integer, loc, denoting whether the beams are in the LEBT, RFQ or MEBT; and nelem, the number of the element in the LEBT in which the particles are located at a given time in the simulation.

```
common /dist/ tdist,loc,nelem
```

Space-Charge Calculations

The standard space-charge subroutine used in RIAPMTQ is SCHEFF. In this subroutine, the particles are distributed over an r-z mesh, and the r- and z-components of the space charge fields are calculated at all the nodes of the mesh. When computing these field components, each particle is assumed to be a circular ring of charge, with the center of each ring lying on the axis passing through the beam centroid. Because the cross section of the beam is in general elliptical rather than circular, the x- and y-components of the fields are adjusted based on the rms values as a first-order correction. SCHEFF can be used for bunched beams and for beam that are in the process of being bunched.

SCHEFF needed to be modified to take into account the multiple charge states. However, a separate space charge subroutine, to be described later, must be used when the approximations made by SCHEFF do not apply. This situation occurs when two charge states pass through a system of bending magnets. The two beams are deflected differently and become separated in the bending plane. The beams cannot be treated separately because they each feel the space-charge forces from the other beam. But they cannot be treated together by SCHEFF because they have different centroids, so the circular approximation is no longer valid. This problem occurs in the LEBT for the RIA designs, where the beams are continuous rather than bunched. In the RFQ and beyond, the beams are bunched and are separated longitudinally, and can be handled satisfactorily by SCHEFF. Even upstream of the RFQ, after the beams go through an achromatic bend in the LEBT, SCHEFF can be used again. After the first buncher cavity, SCHEFF should be used because the beams are starting to be bunched and are no longer continuous in z. It is important that the bending system is achromatic; otherwise, the two beams will not be back on the same axis, and SCHEFF would net be suitable.

The subroutine that handles the continuous beam is called SCHEFXYS. It uses a square mesh in x-y space. It first determines the maximum absolute value of the x- and y-coordinates and makes the outer boundary of the mesh slightly larger, so that all particles will lie inside the mesh. This area is then divided into NB by NB squares, where NB is the number of "bins" in each direction and is an input parameter. (NB is limited to 40.) The charge is distributed on this mesh and, as in SCHEFF, unless the coordinates are in the exact center of one of the squares, the charge is shared among the three nearest neighbors. After the charge is distributed, the x- and y-components of the space charge fields are calculated at all the mesh points and the particles are given an impulse.

Which space-charge routine for PARMTEQ to use, SCHEFF or SCHEFXYS, is indicated by the 9th parameter on the SCHEFF line and on the corresponding TRANS1 5 line (which sets the space-charge parameters). The first line specifying the LEBT elements should be the following:

```
trans1  5  1.  20  0  0  0  0  0  0  -1  0  !set.scheff parameters
```

The first parameter, the "5", tells the PARTRAN subroutine, which handles the beam transport sections before and after the RFQ, to enable the space-charge calculation. The second parameter must be a number greater than zero to indicate there is beam current. The actual current is specified by CURRENT1 and CURRENT2 on the LINAC line. The "-1" in the 9th parameter position tells PARMTEQ to use SCHEFXYS. The only other parameter SCHEFXYS needs is the 3rd parameter, NB, the number of bins. In this example, a 20 x 20 mesh is used. After the beam goes through the first bending magnet, the two charge states separate in the bending plane. One can insert another TRANS1 5 line just before the next bend to increase the number of bins to 30, as shown below.

```
trans1  5  1.  30  0  0  0  0  0  0  -1  0  !set scheff parameters
```

When the beam starts to bunch, SCHEFXYS is no longer applicable, so another TRANS1 5 line is inserted just before the first buncher cavity to tell PARMTEQ to use the normal SCHEFF routine. (NOTE: There may be a problem here if the two beams are not very nearly on the same longitudinal axis, as might occur when there are misalignments.) Below is the standard SCHEFF input line, which is explained in the standard PARMTEQ report[17].

```
trans1  5  1.  .1 .13219 20  40  5  0  0  1  5.2876  !set scheff parameters
```

While the beams are in the process of being bunched in the LEBT, each beam feels the space charge field from the other as well as from itself. So SCHEFF has to treat them together, just as SCHEFXYS does. When the bunched beams arrive at the RFQ, the two charge states are separated longitudinally by 360° at the RFQ frequency, and the bunches of the same charge state are separated by 720° because of the subharmonic buncher. The bunches are assumed to be separated sufficiently that the effect of neighboring bunches does not need to be considered. That is why the 7th parameter, NIP, on the SCHEFF line is zero, whereas it was 5 on the previous TRANS1 5 line. (NIP is the number of identical pulses on each side of the bunch being considered.)

```
scheff  10.  .1 .13219 20  40  0  0  0  1  5.2876
```

Because of the way RIAPMTQ was coded, a SCHEFF line is required if space charge is involved. Before transporting particles through the LEBT, which uses SCHEFXY, the parameters on the SCHEFF line are saved. After completing the SCHEFXY region, the space-charge parameters are restored to the parameters on the SCHEFF line so that SCHEFF can be used.

Several aspects of the multiple charge-state beam-dynamics calculation in the RFQ should be explained. Although the two charge-state bunches are separated by 360° in the RFQ, as far as the RFQDYN subroutine in RIAPMTQ is concerned, they are in the same RFQ cell. Besides, the reference particle arrives at the RFQ at the design phase, so the phases of the other two charge states need to be shifted by 180° to be at the correct phase for being accelerated. The lower charge state arrives late, so the phase of these particles need to be shifted by -180°; the phase of the particles of the higher charge state need to be shifted by +180°. RFQDYN transports both beams through the same cell. But when SCHEFF is called to apply the space-charge impulses, it needs to know the beams are actually separated longitudinally, and to treat each beam as if the other didn't exist, contrary to the way SCHEFF treats the two beams in the LEBT. Therefore, SCHEFF needs to know whether the beams are in the LEBT or in the RFQ or beyond. This is done by a parameter, LOC, in COMMON /DIST/. LOC is set to 1 in the LEBT; to 2 in the RFQ; and to 3 in the MEBT.

Also, when calculating the electric field tables, SCHEFF previously stored the fields produced by one particle, knowing the charge carried by one particle. Since the two beams may carry differing charges per particle, the field tables are for unit charge per particle. The actual charge is included when the charge is distributed on the mesh.

Modifications to IMPACT for the RIA Driver Linac

IMPACT is a parallel particle-in-cell (PIC) beam dynamics code. It has a large collection of beamline elements and calculates the acceleration numerically using RF cavity fields obtained from electromagnetic field-solver codes. It calculates 3D space charge with several boundary conditions. The plan for the RIA version of IMPACT is to introduce new code features including multicharge capability, bending magnets, and stripping models. The multicharge-state capability has already been incorporated and tested. Benchmarking has been carried out against TRACK including comparison of energy gain and rms values. Initial studies have been made of beam loss predictions as well as implementation of bending magnets.

To implement the multiple charge state capability, we have added three attributes to the particle array definition. Each particle array now contains: x, px, y, py, phase, negative energy deviation, charge/mass, current for that charge state, and global particle id. Here, x is normalized by c/omega, c is the speed of light, omega is the 2π times the reference RF frequency, px is normalized by mc, where m is the mass of the particle, y and py are normalized in the same way as x and px, phase is in units of radians, energy deviation is the energy of that particle with respect to reference particle and normalized by mc^2, charge is in the units of electron charge, mass is in the unit of eV, current for that charge state is in units of Amperes, and global particle id is the particle id number associated with the total particles instead of the local particle number.

As a result of this modification, we have updated the particle manager function so that it passes 9 array attributes for each particle instead of only 6 coordinates. We have also updated the particle deposition function to take into account the contributions to the global charge density from each charge state, i.e., each particle has a weight corresponding to the particle attribute 8.

The IMPACT code reads in the spatial dependency of external fields (Ex, Ey, Ez, Bx, By, and Bz) given by a 3D box in Cartesian coordinate, or external fields (Er, Ez, Bthetha)

given by a 2D box in r-z plane (this is based on azimuthal symmetry), Ez on the axis, or Fourier components of Ez on the axis (the other components and radial dependency are inferred from this information).
Given external fields and the space-charge forces, the particles are advanced by solving the Lorentz equations of motion using z as independent variable. The detailed description of these equations can be found in Ref [18]. An implicit leapfrog algorithm is used to solve these equations of motion numerically.

We have also modified the output function to accommodate the 9 attributes of each particle instead of 6. Since the particles may get lost, we have also added two functions to count the lost particles and to update the current from each charge state. As a test of this new version of IMPACT code, we have done particle simulations using a design lattice from ANL as a benchmark. Using an initial waterbag distribution, the simulation results are in general agreement with their results with lower statistical noise due to the larger number of particles used.

Figures 2, 3, and 4 show an IMPACT simulation through a portion of the high-beta linac (81 to 177 MeV) with a comparison with TRACK. Figure 2 shows a plot of rms kinetic energy versus distance along the beam line in the superconducting linac. Figure 3 shows the horizontal rms beam size versus distance. Figure 4 shows the longitudinal rms beam size versus distance. The agreement between IMPACT and TRACK in Figs. 2 and 3 is excellent, and although small differences are observed in Fig.4, the agreement is still good.

The IMPACT code presently achieves near-perfect scalability for RIA modeling. On 256 processors, we estimate that full superconducting linac modeling with five million particles will require about 50 minutes per simulation. Without the factor of 256 increase in speed, a single simulation would require nine days.

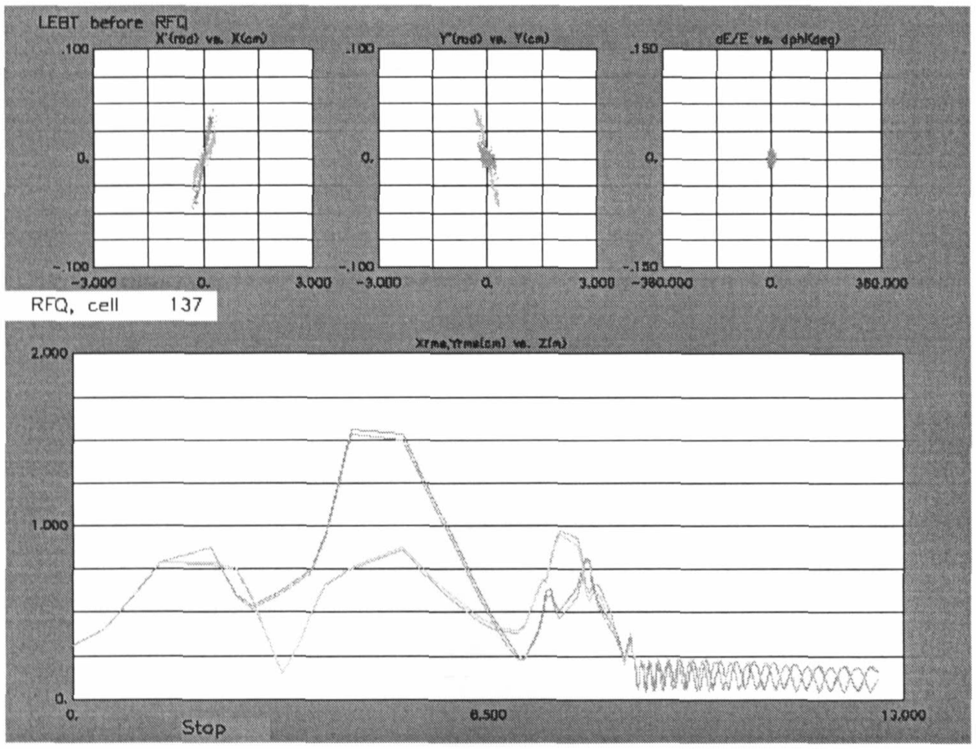

Fig. 1. Three phase space plots (shown at top) at end of the RFQ for a beam with two uranium charge states, q=28 in blue and q=29 in red, for a simulation through a RIA LEBT and RFQ design. Rms beam sizes in x and y are shown at bottom. The red curves in the lower plot show rms values of x for the two charge states, and the blue curves show y.

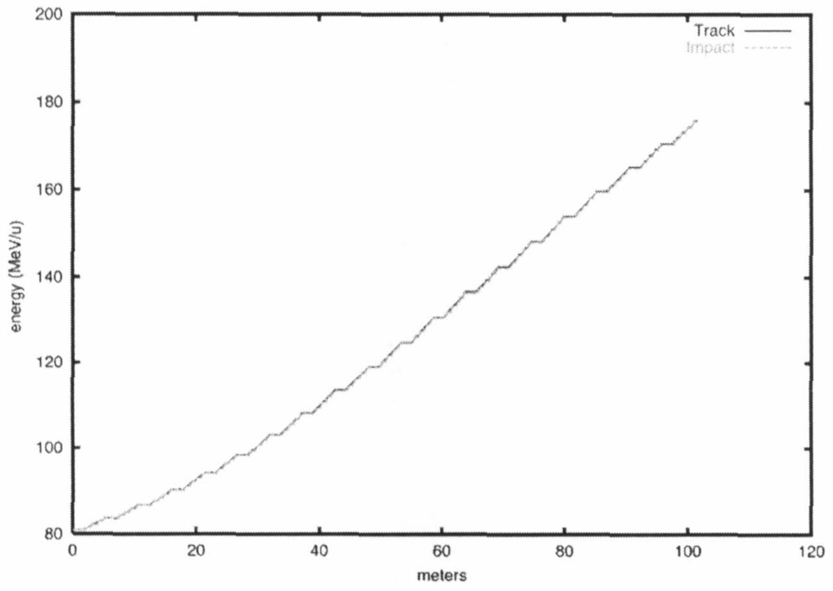

Fig.2. Comparison of IMPACT and TRACK simulations through a portion of the high-beta linac (81 to 177 MeV), showing a plot of rms kinetic energy versus distance along the beam line in the superconducting linac.

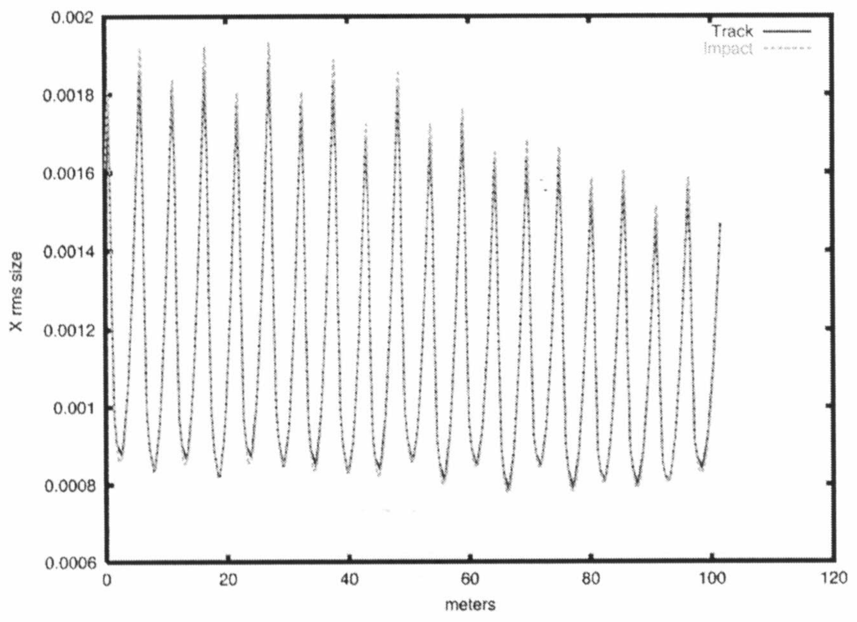

Fig.3. Comparison of IMPACT and TRACK simulations through a portion of the high-beta linac (81 to 177 MeV), showing a plot of horizontal rms beam size versus distance along the beam line in the superconducting linac.

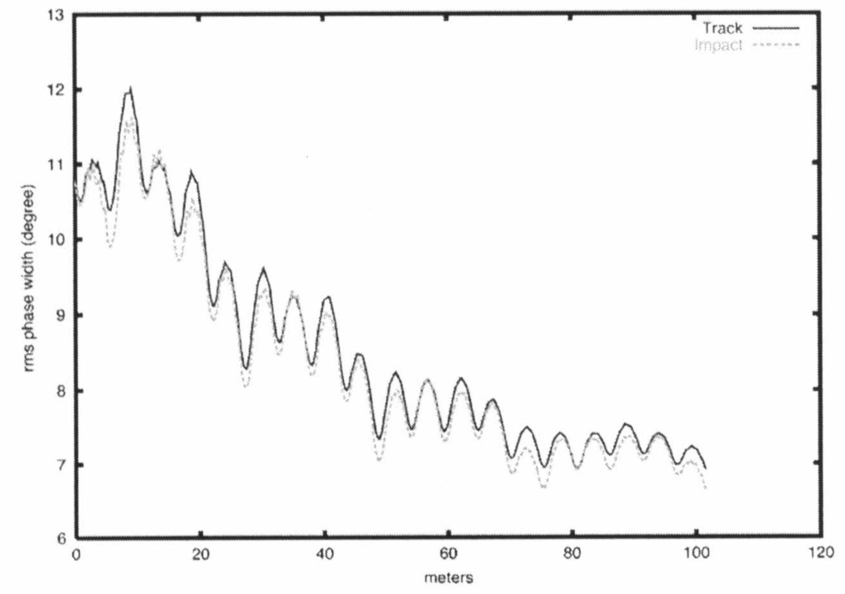

Fig.4. Comparison of IMPACT and TRACK simulations through a portion of the high-beta linac (81 to 177 MeV), showing a plot of longitudinal rms beam size versus distance along the beam line in the superconducting linac.

Development and Status of a Parallel Version of RIAPMTQ/IMPACT at NERSC

As described earlier, a parallel version of PARMTEQ has been developed and installed at NERSC. Our starting point was the Fortran 90 version of PARMTEQ distributed through the LAACG (Los Alamos Accelerator Code Group). Message Passing Interface (MPI) commands were incorporated into this version of PARMTEQ in order to allow use of the code in the parallel-multi-processor environment at NERSC. Optimization of the code using "domain decomposition" was not thought to be necessary. The simpler, more straightforward approach of "particle decomposition" should provide sufficient performance and speed, and was used therefore. Some details of the code modifications required and the benchmarking against simulations on a desktop computer (PC version) are described below.

The first stage of modifying PARMTEQ for parallelization was to strip out all calls to Windows-based graphics and to remove all I/O statements that were machine-environment dependent. To preserve similarity with the PC-based code which was our starting point, identical input file formats were retained. A sample input file (default name RFQ.IN) for two charge states is shown below:

```
run
title
 350.000MHz,q= 1.0,Ws=0.085,Wg=0.300,A=0.265,amu=1.00728,i=70.0mA
trancell
linac 1 0.075 350.000  1.00727647. 1.0
tank 1  6.69 0 0 0 0 0 0 0 0 0 10  0 0 0 0 0 0 0 0 0 0 0 0 0
zdata -5     -3.249    0.010  -90.000  1.000  .0666003
             -2.708    0.054  -90.000  1.000  .0666003
             -2.166    0.386  -90.000  1.000  .0666003
             -1.625    1.092  -90.000  1.000  .0666003
             -1.083    2.006  -90.000  1.000  .0666003
             -0.542    2.783  -90.000  1.000  .0666003
           -0.00001    3.088  -90.000  1.000  .0666003    6
zdata -5      0.000    3.088  -90.000  1.000  .0666003
              8.373    4.104  -90.000  1.015  .0666003
             20.234    5.542  -88.451  1.037  .0666003
             32.096    6.981  -86.902  1.059  .0666003
             43.957    6.981  -85.352  1.080  .0666003
             55.819    6.981  -83.803  1.100  .0666003
             75.413    6.981  -79.511  1.112  .0666003
             87.240    6.981  -75.539  1.125  .0666003
             95.605    6.982  -71.980  1.139  .0666003
            102.044    6.985  -68.813  1.155  .0666003
            107.268    6.991  -65.992  1.171  .0666003
            111.659    6.991  -63.468  1.189  .0666003
            115.444    6.991  -61.199  1.208  .0666003
            118.769    6.991  -59.147  1.217  .0666003
            121.733    6.991  -57.782  1.228  .0666003
            124.408    6.991  -56.578  1.25   .0666003
            126.844    6.991  -55.014  1.263  .0666003
            129.080    6.991  -54.572  1.288  .0666003
            131.147    6.991  -54.239  1.303  0.0668985
            133.069    6.991  -53.500  1.310  0.0675944
            145.85     6.80   -51      1.32   0.0697812
            184.       6.27   -44      1.445  0.0775347
            239.52     5.60   -39      1.600  0.0897613
            300.000    5.000  -35.000  1.750  0.0964213
            360        4.7    -34      1.900  0.0994034
           400.       4.7    -33      1.950  0.0994034
           600        4.0    -34      2.1    0.1059145
           700        3.6125 -37      2.1    0.113
           774       3.2638    -41      1.9784    0.11624
           790.      3.225   -60      1.95    0.117     -1
rfqout 0  4 1
rfqout 01
start 0
stop -1
elimit 2.5
input -6 -5000  1.31 7.90  0.0092  1.31 15.  0.0092 180. 0.
output 3  1 10 00  00 01 1
output 1 1 10 00  00 01 1
output 2 1 00 00  00 01 2 450   2
output 4  1 10 05 .01  1 1
optcon  450  6  0.4  1  0.1  2  60  0  0.2  2
scheff 70.0  0.0274 -0.0271   20 40 5 10  6
exitffl 6.
vfac 1.078
begin
end
```

The definitions of the input quantities given above, and required to run PARMTEQ, are described in the code documentation available through the LAACG. A call to another input file RFQ.PRM was disabled. This file is a control file allowing the user to specify some of the I/O paths for the code execution. These are no longer a user option and the code was modified to either read or write to only the specified files.

PARMTEQ uses some input files generated by our design codes RFQUICK and CURLI which contain detailed design parameters for an RFQ. These files RFQ.TC and RFQ.IMG are typically written as binary files, however, in order to be able to generate these files in the PC environment and then transfer them to NERSC for use in large-scale simulations, it was necessary to convert these files to ASCII format. Future modification of the design codes to directly write these files in ASCII format will make transfer of these files and use of these files at NERSC more direct.

The ability to write output information for a single-processor simulation was preserved in order to be able to make direct comparisons with results obtained on the desktop PC. The single-processor PC-style output is generated in subroutine OUTPUTK and incorporates all additions and changes made to the desk-top version. This output is now written to the file named "outfile" at NERSC. However, in order to produce similar output for multi-processor simulations, a new output subroutine OUTRMS2 was required which gathers all the multi-particle coordinate information from all the processors and calculates rms beam properties using the summed information. Output from this subroutine is written to the file parmteqm.out. By running in single-processor mode at NERSC we were able to check our results against those produced by the desk-top PC. Agreement to within a few percent was observed, with the discrepancy being attributed to slightly different initial particle distributions resulting from differences in random number generation. Debugging of the OUTRMS2 subroutine was accomplished by comparing its output with that of OUTPUTK for a single-processor simulation. Both output files contain rms beam properties as a function of distance along the transport and accelerator sections for each charge state (up to two beams) plus those of the combined beam distributions. It is expected that the combined distribution information will be useful to understand the acceptances of the system and to simulate the effects of beam steering algorithms.

In addition to generating output files similar in format to the desk-top PC version of the code, several additional output files are generated. Three output files, one for charge state 1 (fort.28), one for charge state 2 (fort.29), and one for the combined beam parameters (fort.30) are generated in tabular form to allow plotting with a post-processor graphics program later. Finally, an ASCII output file containing the 6 phase space coordinates of each macro-particle is generated for each charge state. These two files are named dist1.dat and dist2.dat. It is these output distribution files that will be read in by the IMPACT code.

The most significant code modifications were required in the parallelization of the space-charge calculations which consume the majority of the computing time in multi-particle simulations. In order to speed up the calculations, each processor generates its own initial particle distributions (a subset of the total distribution) for each charge state. Each sub-distribution propagates independently of the other. All the particles only interact via the space-charge forces. Through the appropriate MPI calls, each particle is given the appropriate space-charge impulse or particle quantities are summed by calls to the output subroutines.

When the space-charge subroutines SCHEFXYS or SCHEFF are called, each processor owns a mesh upon which its own charge has been deposited. A call to MPI_ALLREDUCE is used to sum up all the charges and distribute the total charge density to all processors. Each processor can now correctly calculate the fields at its mesh points and give the correct space-charge impulses to its particles. Otherwise, the same identical mesh algorithms and approximations are used as in the PC version.

As is to be expected, additional modifications in variable declarations, variable initializations, and common blocks were required to run at NERSC. However, it is expected that most for our implementations should be general enough to make this code easily portable to other multi-processor platforms, if necessary.

As part of the debugging and benchmarking processes, several simulations were run using an example input file shown above while varying the number of macro-particles, the number of nodes, and the number of processors. Simulation results at the exit of the sample RFQ were compared for a single processor on a single node up to 256 processors (16 processors on 16 nodes). Variations in transverse output emittances of up to 11% were observed and variations as large as 22% in the output longitudinal emittances were observed. However, average variations in transmission were less than 1% as a function of the number of processors used. Some numerical noise is expected. A saturation effect should be observed as the number of macro-particles is increased, whereby significant variations in the rms beam properties as a function of the number of particles diminish.

Conclusions

The objective of the project is to develop an end-to-end multiparticle parallel beam-dynamics simulation tool called RIAPMTQ/IMPACT suitable for high-statistics computation of beam dynamics and beam losses in the RIA driver linac. This report describes the progress during the first year supported by the FY03 funding. This work includes development of a multiparticle parallel version of PARMTEQ to model the multicharge-state beam dynamics, including space charge, in the critical low-velocity region of the linac that includes the low-energy beam transport (LEBT), the radiofrequency quadrupole (RFQ), and the medium-energy transport section that matches the beam into the main superconducting linac. It also includes work [19] to develop the IMPACT simulation code for the superconducting part of the RIA Driver Linac.

We have installed a version of the PARMTEQ/IMPACT code at the NERSC supercomputer, in addition to a version that runs on the desktop computer. Future work will include addition of stripper subroutines, implementation of error or off-normal conditions conditions, a systematic benchmarking of both the modified desktop and NERSC versions against the rms results of the TRACK and LANA codes.

REFERENCES

[1] K.W.Shepard, J.R.Delayen, C.N.Lyneis, J.Nolen, P.Ostroumov, J.W.Staples, J.Brawley, C.Hovater, M.Kedzie, M.P.Kelly, J.Mammosser, C.Piller, and M.Portillo, "SC Driver Linac for a Rare Isotope Facility," Proceedings of the 9th Workshop on RF Superconductivity, Santa Fe, NM, 1999, ed. by B. Rusnak (LANL, Los Alamos, 1999) Vol. 2, p.345 ; ISOL Task Force Information including the "ISOL Task Force Report to NSAC", November 22, 1999 (available at http://srfsrv.jlab.org/ISOL/); C.W.Leemann,"The Rare-Isotope Accelerator (RIA) Facility Project," Proc. XX International Linac Conference, Monterey, CA, August 21-25, 2000, p. 331.
[2] P.N.Ostroumov and K.W.Shepard, "Multiple-Charge Beam Dynamics in an Ion Linac," Phys. Rev. ST Accel. Beams 3, 030101 (2000).
[3] P.N.Ostroumov, R.C,Pardo, G.P.Zinkann, K.W.Shepard, and J.A.Nolen, "Multiple Charge State Beam Acceleration at Atlas," Phys. Rev. Lett. 86, 2798-2801 (2001).
[4] P. N. Ostroumov, "Heavy-Ion Beam Dynamics in the Rare Isotope Accelerator Facility", presented at 2003 Particle Accelerator Conf., Portland, OR, May 12-16, 2003.
[5] P. N. Ostroumov, "Development of a Medium-Energy Superconducting Heavy-Ion Linac", Phys. Rev. ST Accel. Beams 5, 030101 (2002).
[6] P. N. Ostroumov, "Sources of Beam Halo Formation in Heavy-Ion Superconducting Linac and Development of Halo Cleaning Methods", presented at the HALO-03 IVFA Advanced Beam Dynamics Workshop, Montauk, NY, May 19-23, 2003.
[7] D. Gorelov, T.L.Grimm, W. Hartung, F. Marti, X. Wu, R.C.York, H. Podlech, "Beam Dynamics Studies at NSCL of the RIA Superconducting Driver Linac," Proc. European Particle Accelerator Conf., Paris, France, 2002, p. 900.
[8] D.V.Gorelov, and P.N.Ostroumov, "Application of LANA Code for Design of Ion Linac," Proc. of European Particle Accelerator Conf. 1996, Barcelona, Spain, Vol.2, 1996, p. 1271.
[9] P. N. Ostroumov and K. W. Shepard, "Correction of Beam-Steering Effects in Low-Velocity Superconducting Quarter-Wave Cavities", Phys. Rev. ST Accel. Beams 4, 110101 (2001).
[10] D.V.Gorelov, P.N.Ostroumov, and R.E.Laxdal, "Use of the LANA Code for the Design of a Heavy Ion Linac," Proc. of 1997 Particle Accelerator Conf. Vancouver Canada, 1998, p. 2621.
[11] D.V.Gorelov and P.N.Ostroumov, "Simulation of Beam Dynamics Including Space Charge in Proton Linac with Errors," Proc. of XIX International Linac Conf., Chicago, IL, 1998, p.654.

[12] X. Wu, D. Gorelov, T.L.Grimm, W. Hartung, F. Marti, R.C.York, "The Misalignment and RF Error Analyses for the RIA Driver Linac," presented at XXI International Linear Accelerator Conf. LINAC 2002, Gyeongju, Korea, August 19-23, 2002.
[13] B. Mustapha, J. A. Nolen, and P. Ostroumov, "Large Scale Computing for Beam Dynamics Simulations and Target Modeling," RIA Research and Development Workshop, Aug. 26-28, 2003, Bethesdat, MD.
[14] K.W.Shepard et al., "The RIA Driver Linac," presented at XXI International Linear Accelerator Conf. LINAC 2002, Gyeongju, Korea, August 19-23, 2002.
[15] K. R. Crandall and T. P. Wangler, "PARMTEQ - Beam Dynamics Code for the RFQ Linear Accelerator", AIP Conf. Proc. 177, Linear Accelerator and Beam Optics Codes, C. Eminhizer, ed., 1988, pp. 22-28.
[16] J. Qiang, R. D. Ryne, S. Habib, and V. Decyk, J. Comput. Phys. **163**, 434 (2000).
[17] K. R. Crandall, T. P. Wangler, L. M. Young, J. H. Billen, G. H. Neuschaefer, and D. L. Schrage, Documentation by J. H. Billen, Los Alamos National Laboratory, LA-UR-96-1836, Revised December 6, 2000.
[18] .J.Qiang et. al, "Macroparticle Simulation Studies of a Proton Beam Halo Experiment", PRSTAB, vol.5 124201 (2002).
[19] R. Ryne et al., Advanced Beam-Dynamics Simulation Tools for the RIA Driver Linac/Part 1: Supercond-ucting Linac, these proceedings.